Luke Mueller
lam908@mail.harvard.edu
CS181-S16

Assignment #3

Due: 5:00pm March 25, 2016

Collaborators:

Homework 3: SVM

There is a mathematical component and a programming component to this homework. Please submit ONLY your PDF to Canvas, and push all of your work to your Github repository. If a question asks you to make any plots, like Problem 3, please include those in the writeup.

Problem 1 (Fitting an SVM by hand, 8pts)
Consider a dataset with the following 6 points in $1D$:

$$\{(x_1, y_1)\} = \{(-3, +1), (-2, +1), (-1, -1), (1, -1), (2, +1), (3, +1)\}$$

Consider mapping these points to 2 dimensions using the feature vector $\phi : x \mapsto (x, x^2)$. The max-margin classifier objective is given by:

$$\min_{\mathbf{w}, w_0} \|\mathbf{w}\|_2^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^T \phi(x_i) + w_0) \geq 1, \ \forall i \tag{1}$$

Note: the purpose of this exercise is to solve the SVM without the help of a computer, relying instead on principled rules and properties of these classifiers. The exercise has been broken down into a series of questions, each providing a part of the solution. Make sure to follow the logical structure of the exercise when composing your answer and to justify each step.

1. Write down a vector that is parallel to the optimal vector $\mathbf{w}$. Justify your answer.
2. What is the value of the margin achieved by $\mathbf{w}$? Justify your answer.
3. Solve for $\mathbf{w}$ using your answers to the two previous questions.
4. Solve for w_0. Justify your answer.
5. Write down the discriminant as an explicit function of x.

Solution

1. First, we note that $\mathbf{w}$ is orthogonal to the decision boundary given by $y_i(\mathbf{w}^T\phi(x_i) + w_0) = 0$ since for two points x_1 and x_2 on the decision boundary, $(x_1 - x_2) * \mathbf{w} = -w_0 - (-w_0) = 0$. Then, considering $x_\perp$, which is defined as the vector on the decision boundary to satisfy the equation $\phi(x_\perp) + r\frac{\mathbf{w}}{||\mathbf{w}||_2} = \phi(x_i)$ for some point $\phi(x_i)$ in our data, we know that $\phi(x_i) - \phi(x_\perp)$ will also be orthogonal to the decision boundary. Thus, it follows that $\mathbf{w} \parallel \phi(x_i) - \phi(x_\perp)$.

   For example, taking the point $\phi(x_i) = ((2, 4), +1)$ whose respective $\phi(x_\perp) = ((2, 2.5), 0)$:

   $$\mathbf{w} \parallel \phi(x_i) - \phi(x_\perp) = ((2, 4), +1) - ((2, 2.5)0) = (0, 1.5)$$

2. The value of the margin achieved by $\mathbf{w}$ is easily seen in a $2D$ diagram of $\phi(x)$, since the support vectors are made obvious. In particular, for $y_2 = +1$ the support vectors are $((-2, 4), +1)$ and $((2, 4), +1)$ and for $y_2 = -1$ the support vectors are $((-1, 1), -1)$ and $((1, 1), -1)$. We know this because (1): they are the closest, oppositely classified vectors in the data, and (2): the difference between these

positively classified vectors is parallel to the difference in these negatively classified vectors, and thus, also parallel to the optimal decision boundary. Then, placing the decision boundary equidistant from these 4 support vectors, we can calculate the margin achieved by $\mathbf{w}$ as the distance between any support vector and the decision boundary, which comes out to 1.5.

3. To determine $\mathbf{w}$, we make use of the following:

$$\phi(x_\perp) + r\frac{\mathbf{w}}{||\mathbf{w}||_2} = \phi(x_i)$$
$$r = \frac{1}{||\mathbf{w}||_2}(\phi(x)\mathbf{w}^T + w_o)$$
$$r = 1.5$$

The first equation we defined previously. The second is the geometric margin, and we make use of the fact that $\mathbf{w}$ is determined when the functional margin $(\phi(x)\mathbf{w}^T + w_o)$ equals 1. The third equation is given by the solution to (2.) Thus we have:

$$\phi(x_\perp) + r\frac{\mathbf{w}}{||\mathbf{w}||_2} = \phi(x_i) \Rightarrow \frac{\mathbf{w}}{||\mathbf{w}||_2} = \frac{\phi(x_i) - \phi(x_\perp)}{r} = \frac{(2,4) - (2,2.5)}{1.5} = \frac{(0,1.5)}{1.5}$$

Then solving for $||\mathbf{w}||_2$:

$$r = \frac{1}{||\mathbf{w}||_2}(\phi(x)\mathbf{w}^T + w_o) = \frac{1}{||\mathbf{w}||_2} \Rightarrow ||\mathbf{w}||_2 = \frac{1}{r} = \frac{1}{1.5} = 2/3$$

Then solving for $\mathbf{w}$

$$\mathbf{w} = \frac{\phi(x_i) - \phi(x_\perp)}{r} \times ||\mathbf{w}||_2 = \frac{(0,1.5)}{(1.5)^2} = (0,2/3)^T$$

4. To solve for w_0 we input $\mathbf{w}$ and some point $\phi(x_i)$ into the functional margin. In particular we have:

$$(\phi(x)\mathbf{w}^T + w_o) = 1 \Rightarrow w_0 = 1 - \phi(x)\mathbf{w}^T = 1 - (2,4)\cdot(0,2/3) = 1 - (0 + 8/3) = -1\frac{2}{3}$$

5. Using values for $\mathbf{w}$ and w_0 we write the discriminant as an explicit function of x:

$$y_i = \mathbf{w}^T\phi(x_i) + w_0 = (0,\frac{2}{3})^T \cdot (x_i, x_i^2) - 1\frac{2}{3} \tag{2}$$

Problem 2 (Composing Kernel Functions , 7pts)
Prove that

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\{-||\boldsymbol{x} - \boldsymbol{x}'||_2^2\}\,,$$

where $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^D$ is a valid kernel, using only the following properties. If $K_1(\cdot,\cdot)$ and $K_2(\cdot,\cdot)$ are valid kernels, then the following are also valid kernels:

$$\begin{aligned}
&(1) \quad K(\boldsymbol{x}, \boldsymbol{x}') = c\, K_1(\boldsymbol{x}, \boldsymbol{x}') \quad \text{for } c > 0 \\
&(2) \quad K(\boldsymbol{x}, \boldsymbol{x}') = K_1(\boldsymbol{x}, \boldsymbol{x}') + K_2(\boldsymbol{x}, \boldsymbol{x}') \\
&(3) \quad K(\boldsymbol{x}, \boldsymbol{x}') = K_1(\boldsymbol{x}, \boldsymbol{x}')\, K_2(\boldsymbol{x}, \boldsymbol{x}') \\
&(4) \quad K(\boldsymbol{x}, \boldsymbol{x}') = \exp\{K_1(\boldsymbol{x}, \boldsymbol{x}')\} \\
&(5) \quad K(\boldsymbol{x}, \boldsymbol{x}') = f(\boldsymbol{x})\, K_1(\boldsymbol{x}, \boldsymbol{x}')\, f(\boldsymbol{x}') \quad \text{where } f \text{ is any function from } \mathbb{R}^D \text{ to } \mathbb{R}
\end{aligned}$$

Solution

First, we expand the square $||\mathbf{X} - \mathbf{X}'||_2^2$ as in Bishop 6.24 to get:

$$\mathbf{X}^T\mathbf{X} + (\mathbf{X}')^T\mathbf{X}' - 2\mathbf{X}^T\mathbf{X}'$$

Then multiplying by -1 and exponentiating we obtain:

$$\exp\{-\mathbf{X}^T\mathbf{X}\}\exp\{2\mathbf{X}^T\mathbf{X}'\}\exp\{-(\mathbf{X}')^T\mathbf{X}'\}$$

Which is a valid kernel due to:

1. The validity of the linear kernel $k(\mathbf{X}, \mathbf{X}') = \mathbf{X}^T\mathbf{X}$, which is present in all 3 terms.
2. The validity of a kernel multiplied by a scalar to account for the 2 in the second term
3. The validity of the exponent of a valid kernel, given by (4) above, which applies to all 3 terms.
4. The validity of a kernel in the form given by (5) above, where the first and third terms are a function of $\mathbf{X}$ and the second term, $\exp\{2\mathbf{X}^T\mathbf{X}'\}$, is equivalent to $K_1(\boldsymbol{x}, \boldsymbol{x}')$ in (5).

Problem 3 (Scaling up your SVM solver, 10pts (+7pts with extra credit))

In the previous homework, you studied a simple data set of fruit measurements. We would like you to code up a few simple SVM solvers to classify lemons from apples. To do this, read the paper at http://www.jmlr.org/papers/volume6/bordes05a/bordes05a.pdf and implement the Kernel Perceptron algorithm and the Budget Kernel Perceptron algorithm. The provided code has a base Perceptron class, which you will inherit to write KernelPerceptron and BudgetKernelPerceptron. This has been set up for you in problem3.py. The provided data is linearly separable. Make the optimization as fast as possible.

Additionally, we would like you to do some experimentation with the hyperparameters for each of these models. Try seeing if you can identify some patterns by changing β, N (maximum number of support vectors), or the number of random samples you take. Note the training time, accuracy, shapes/orientations of hyperplanes, and number of support vectors for various setups. We are intentionally leaving this open-ended to allow for experimentation, and so we will be looking for your thought process and not a rigid graph this time. That being said, any visualizations that you want us to grade and refer to in your descriptions should be included in this writeup. You can use the trivial $K(\mathbf{x_1}, \mathbf{x_2}) = \mathbf{x_1}^T\mathbf{x_2}$ kernel for this problem, though you are welcome to experiment with more interesting kernels too. Also, answer the following reading questions in one or two sentences each.

1. In one short sentence, state the main purpose of the paper?
2. Identify each of the parameters in Eq. 1
3. State one guarantee for the Kernel perceptron algorithm described in the paper.
4. What is the main way the budget kernel perceptron algorithm tries to improve on the perceptron algorithm.
5. In simple words, what is the theoretical guarantee of LASVM algorithm? How does it compare to its practical performance?

For extra credit (+7 pts), implement the SMO algorithm and implement the LASVM process and do the same as above.

Solution

1. The main purpose of the paper is to explore properties of various kernel classifiers in order to determine whether certain "examples" in our data should be given additional computing time.
2. Eq. 1: $\hat{y}(x) = w'\boldsymbol{\phi}(x) + b$
   - $\hat{y}(x)$ is the predicted classification for input value x.
   - w' is a parameter vector that maps input x onto its predicted classification.
   - $\boldsymbol{\phi}(x)$ is the feature vector transforming x.
   - b is the bias parameter that displaces $w'\boldsymbol{\phi}(x)$ from the origin.
3. One guarantee of the Kernel perceptron algorithm described in the paper is "When a solution exists, Novikoffs Theorem (Novikoff, 1962) states that the algorithm converges after a finite number of mistakes, or equivalently after inserting a finite number of support vectors".
4. The budget kernel perceptron algorithm attempts to improve on the perceptron algorithm by adding a step to remove support vectors from the kernel expansion, thus setting an upper bound to the number of support vectors generated by the algorithm. In particular, it introduces a hyperparameter β instead of 0 for the condition $y_t\hat{y}(x_t) \leq \beta$, and adds an additional condition, "If $|S| > N$ then $S \leftarrow S - \{\arg\max_{i\epsilon S} y_i(\hat{y}(x_i) - \alpha_i K(x_i, x_i))\}$".

5. The theoretical guarantee of the LASVM algorithm is that it converges to the solution of the SVM QP (Quadratic Programming) problem after sufficient sequential passes over the training examples. However, experimental evidence indicates that LASVM matches the SVM accuracy after a single sequential pass over the training examples (page 1585). In other words, if we run the LASVM algorithm enough times we will arrive at the SVM solution, but practically speaking, we only need to run LASVM once to get a sufficient solution.

Problem 3 Results

Table 1: Hyperparameter Tuning

Model	Beta	N	Numsamples	# Support Vectors	Process Time (s)	Accuracy
model 1 k	0	100	20000	13	0.98481	1
model 1 bk	0	100	20000	100	9.364486	0.9999392633
model 2 k	0	200	20000	184	6.807833	1
model 2 k	0	200	20000	200	19.112219	1
model 3 k	0	500	20000	212	8.086729	1
model 3 bk	0	500	20000	70	6.900058	1
model 4 k	0	100	10000	158	3.531878	0.9998785265
model 4 bk	0	100	10000	100	4.804197	0.9620395396
model 5 k	0	100	30000	103	8.50177	1
model 5 bk	0	100	30000	100	15.577188	1
model 6 k	0	100	50000	200	23.909554	1
model 6 bk	0	100	50000	99	25.03202	0.9875793374
model 7 k	0.5	100	20000	87	5.084028	1
model 7 bk	0.5	100	20000	100	10.82751	0.9806553494
model 8 k	0.05	100	20000	231	8.934602	0.9999392633
model 8 bk	0.05	100	20000	100	10.938737	0.9921345926
model 9 k	-0.05	100	20000	280	9.548529	1
model 9 bk	-0.05	100	20000	0	0.268766	0
model 10 k	0.05	200	10000	236	4.174034	0.9856964985
model 10 bk	0.05	200	10000	200	8.645547	0.9712411552
model 11 k	0.05	500	10000	138	3.240456	0.9999392633
model 11 bk	0.05	500	10000	406	11.599665	0.9861823924

Table 1 shows the relative performances of the Kernel Perceptron algorithm (k) and the Budget Kernel Perceptron algorithm (bk) under different hyperparameter values. In particular, we have:

- *beta*: the missclassification condition
- *N*: the upper bound for support vectors. Applied only in Budget Kernel Perceptron
- *numsamples*: the number of examples taken from the training data to execute the algorithms. Applied only in Budget Kernel Perceptron

The process time is represented in seconds. The accuracy was determined by splitting the data into training and validation sets, fitting the training data with the algorithm, using the fit to predict new values for the validation data, and computing the proportion of correctly classified validation predictions.

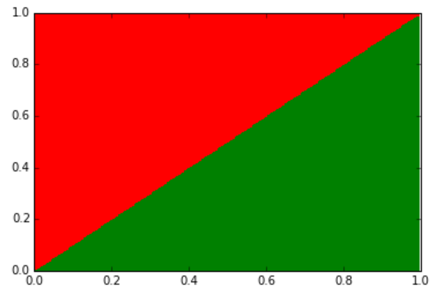

Figure 1: Kernel Perceptron.

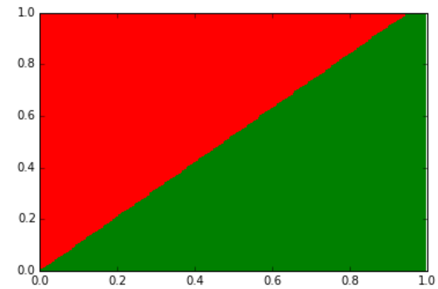

Figure 2: Budget Kernel Perceptron.

In general, all models provided near perfect accuracy with varied process time across different values for the hyperparameters. As expected, increasing the number of samples provided the best accuracy with the greatest cost of process time. Increasing the upper bound for the support vectors also increased process time, but did not necessarily improve the accuracy. Changing beta decreased accuracy as it deviated further from 0.

Notably, though the accuracy metric reveals slight differences between models, relatively speaking there were no substantial differences.

The Kernel Perceptron algorithm generally ran faster, and provided greater accuracy than the Budget Kernel Perceptron algorithm. Model 3 was an exception, where the upper bound on the number of support vectors was large.

The shapes/orientations of the hyperplanes were slightly different under some hyperparameter settings, as seen in figures 1 and 2. They were identical however, when *numsamples* was large - since the support vectors were not as dominated by the algorithms' random example selections.

To summarize, the Kernel Perceptron algorithm generally performed better than the Budget Kernel Perceptron algorithm with regards to training process time and validation accuracy. This is likely due to the fact that our data are not very "noisy", thus the benefits provided by the Budget Kernel Perceptron algorithm are somewhat lost. Concretely, the increased margin provided by the modified decision boundary under the Budget Kernel Perceptron algorithm was unnecessary, and possibly even over-fitted while costing us additional computation time.

Calibration [1pt]

Approximately how long did this homework take you to complete?
15 hours.